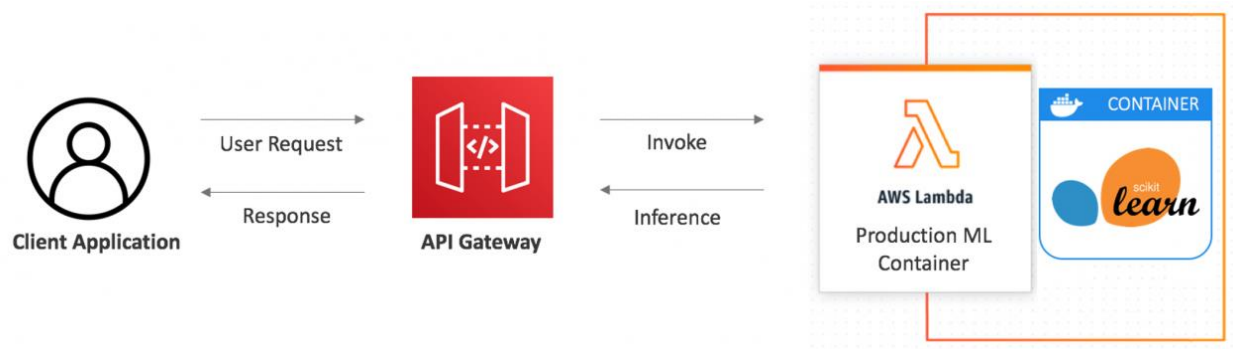
Python/ Deep Learning Programming Final Project

Title: Serverless Deep Learning Framework

Team Name: Winter is Coming

Members: Srinivas Rahul Sapireddy (Individual Submission)

Introduction



Here we introduce a clothes classification service. Where the user takes the picture of the clothes they want to wear. Then the classifier tells the user you are trying to buy in the clothes category.

Here I have used keras and tensorflow for image classification for classifying the pictures of different clothes.

Here I have used AWS Lambda for deploying the model. Lambda is a service from AWS which allows us to deploy many different machine learning models.

Problem Statement

Here we send the picture with the URL to the model that we are going to deploy with AWS Lambda and then the service will reply with different classes and we will also have some score for it.

Here we use tensorflow-lite internally.

Questions Answered?

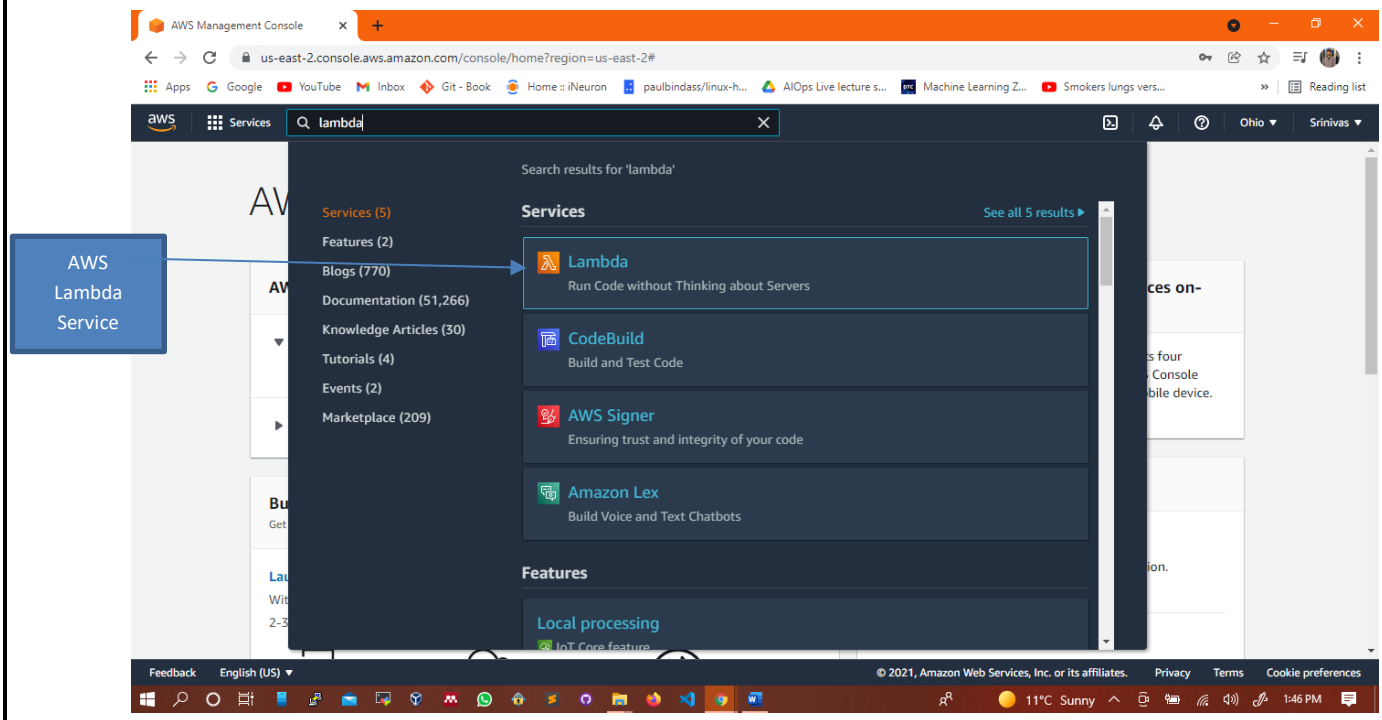
What AWS Lambda is and difference between other approaches?

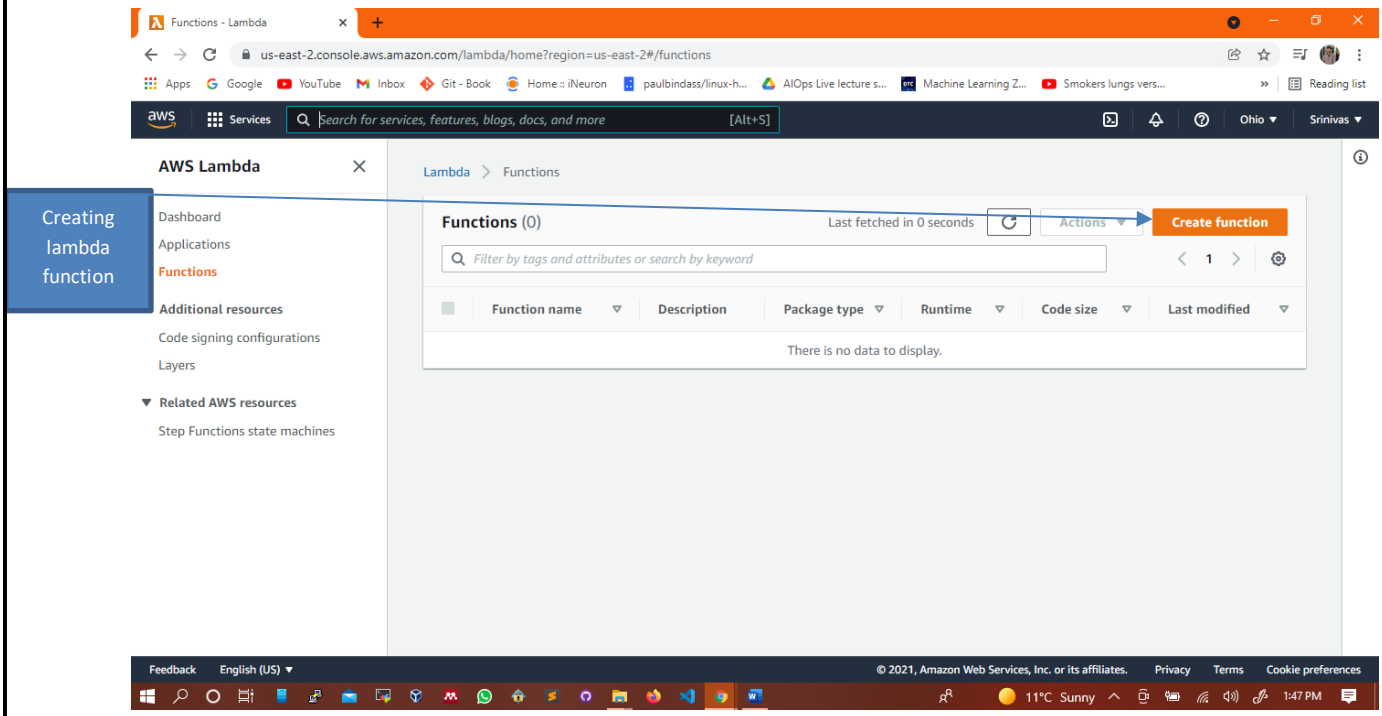
What is the tensorflow-lite and why it is better for this particular use case?

We will package everything in to a docker container and deploy it to AWS Lambda. Finally, we will expose this Lambda function as web service API gateway.

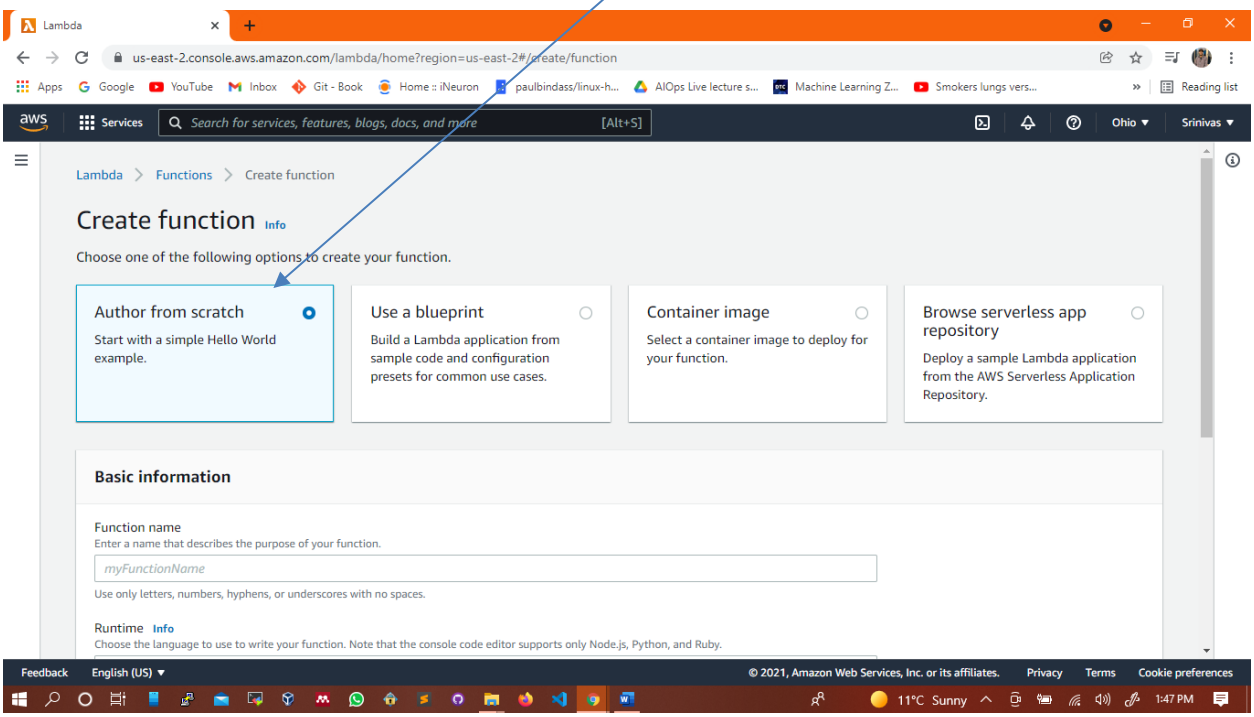
Serverless is the concept of removing infrastructure considerations for deploying code. Instead of having to manage servers and infrastructure by ourselves, a serverless service takes care of that for us and only charges according to use.

AWS LAMBDA

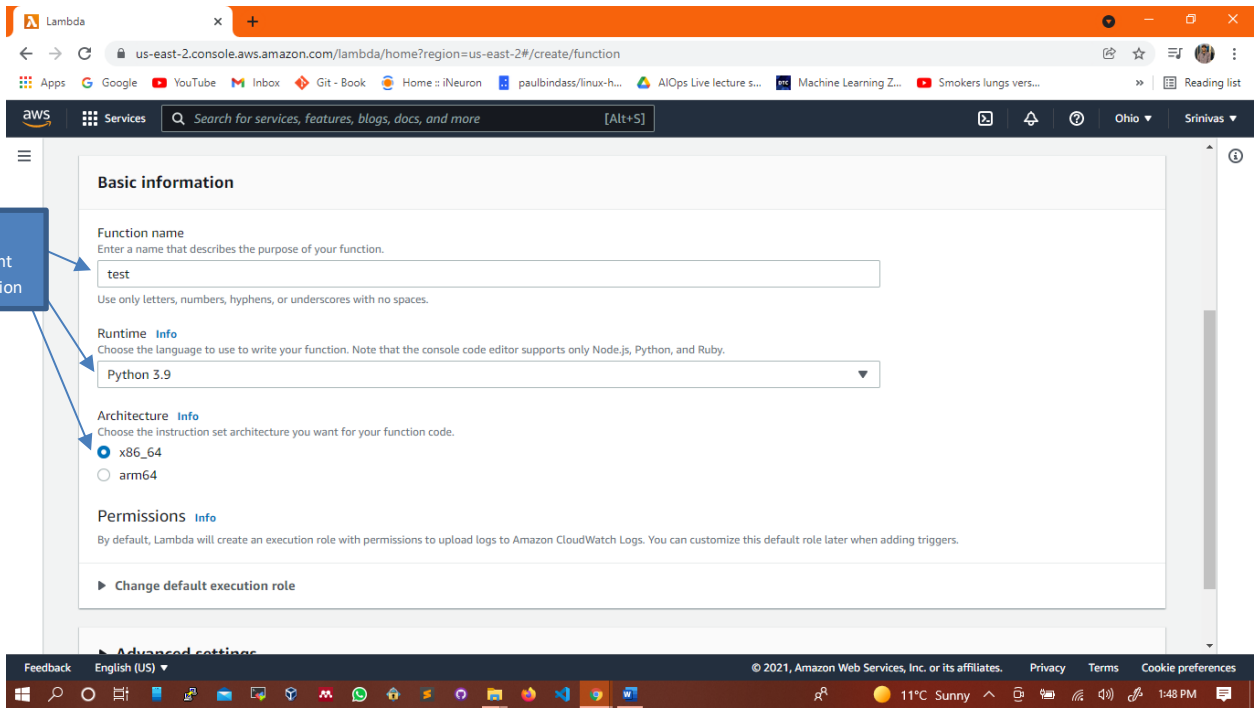


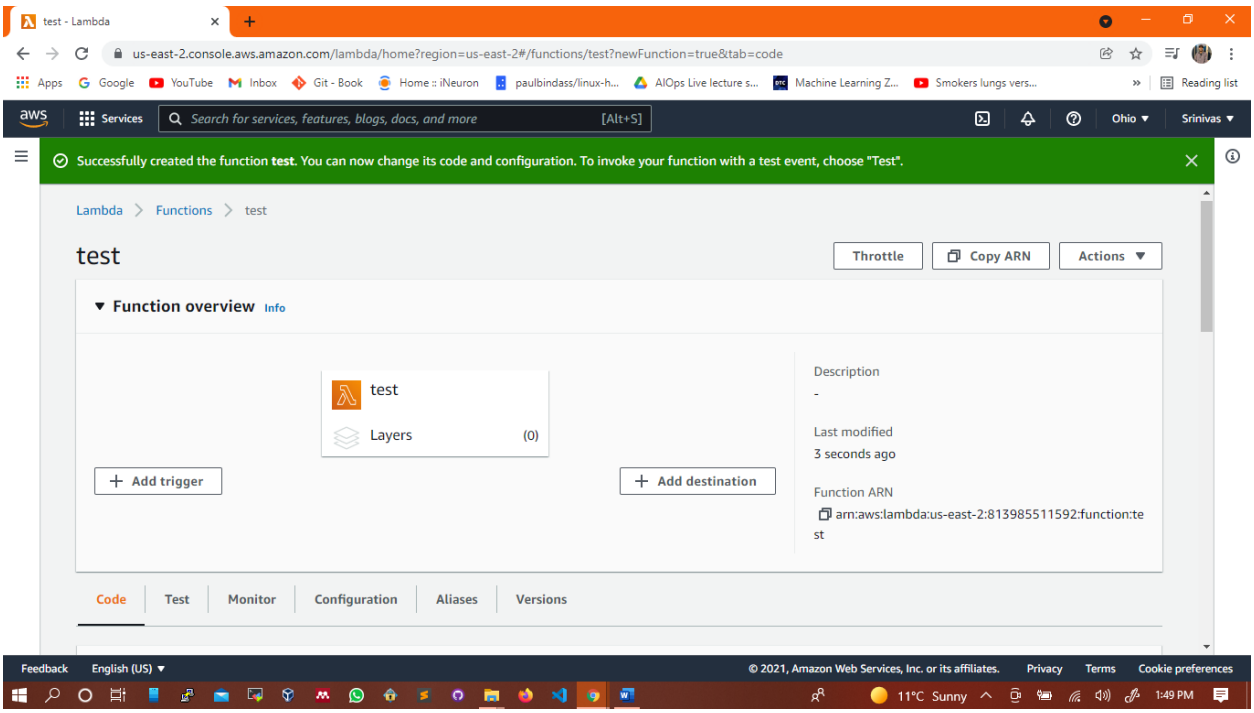


Creating lambda function

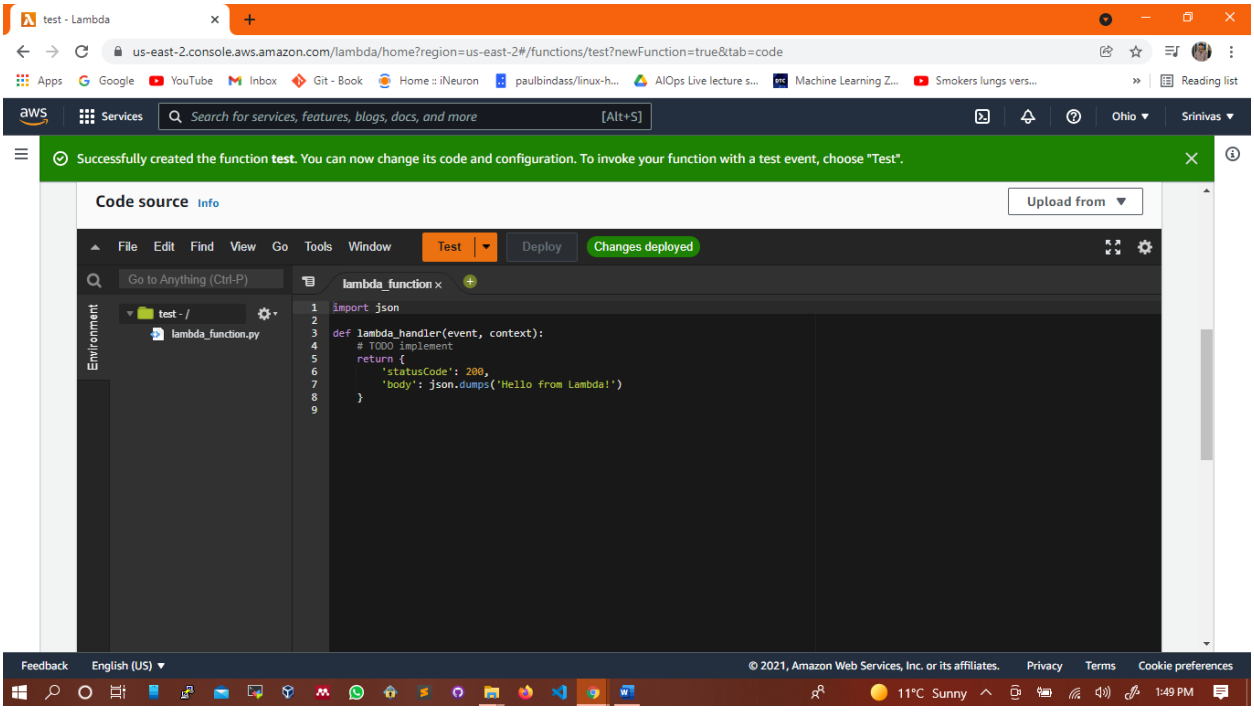


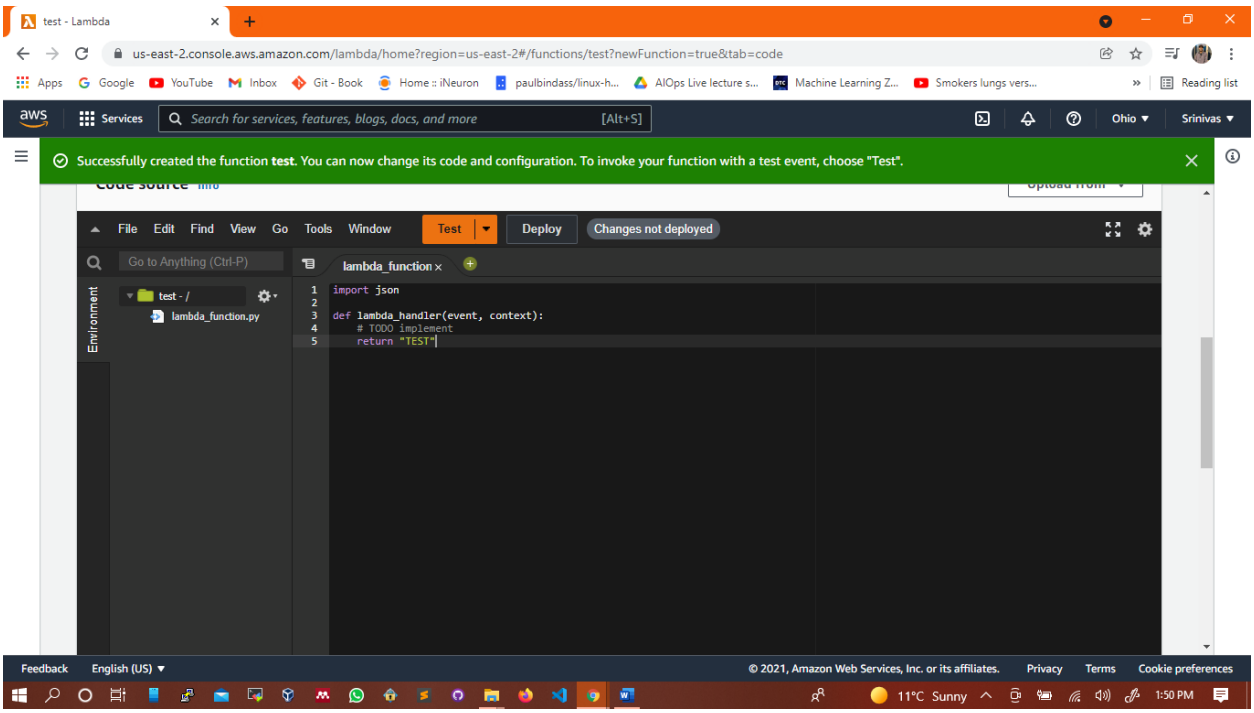
Selecting Environment for the function





We can see the code of this function here.

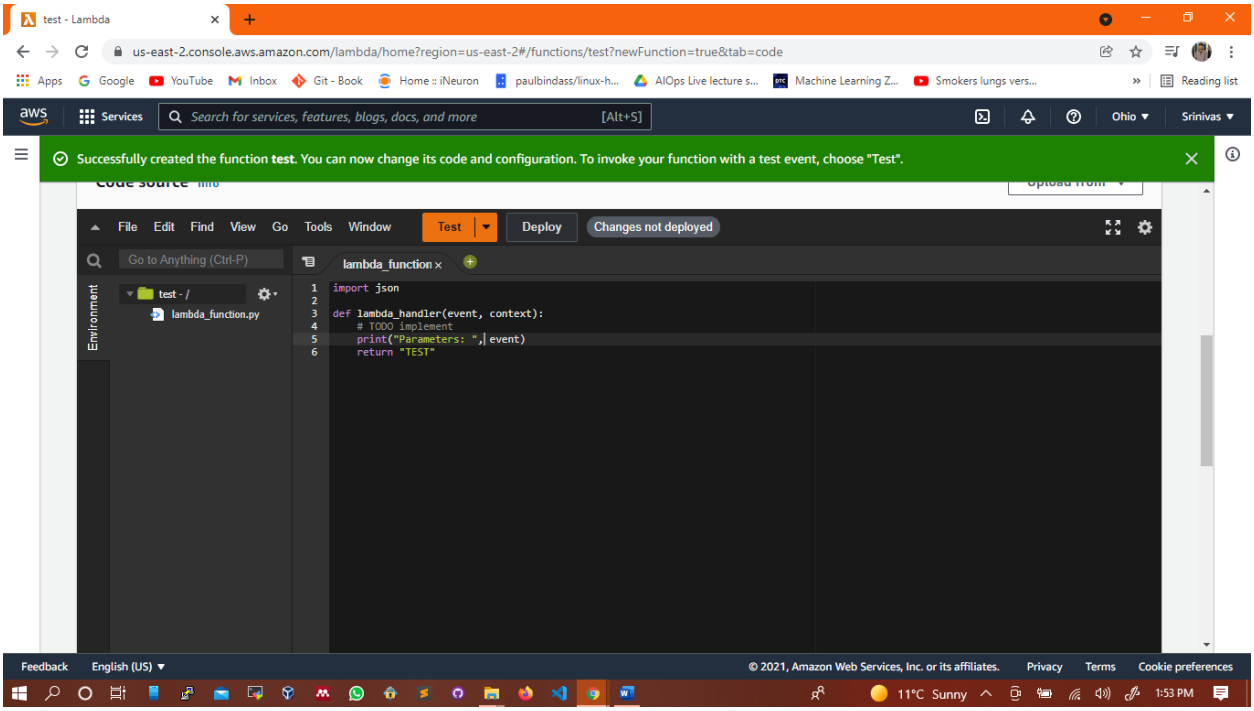




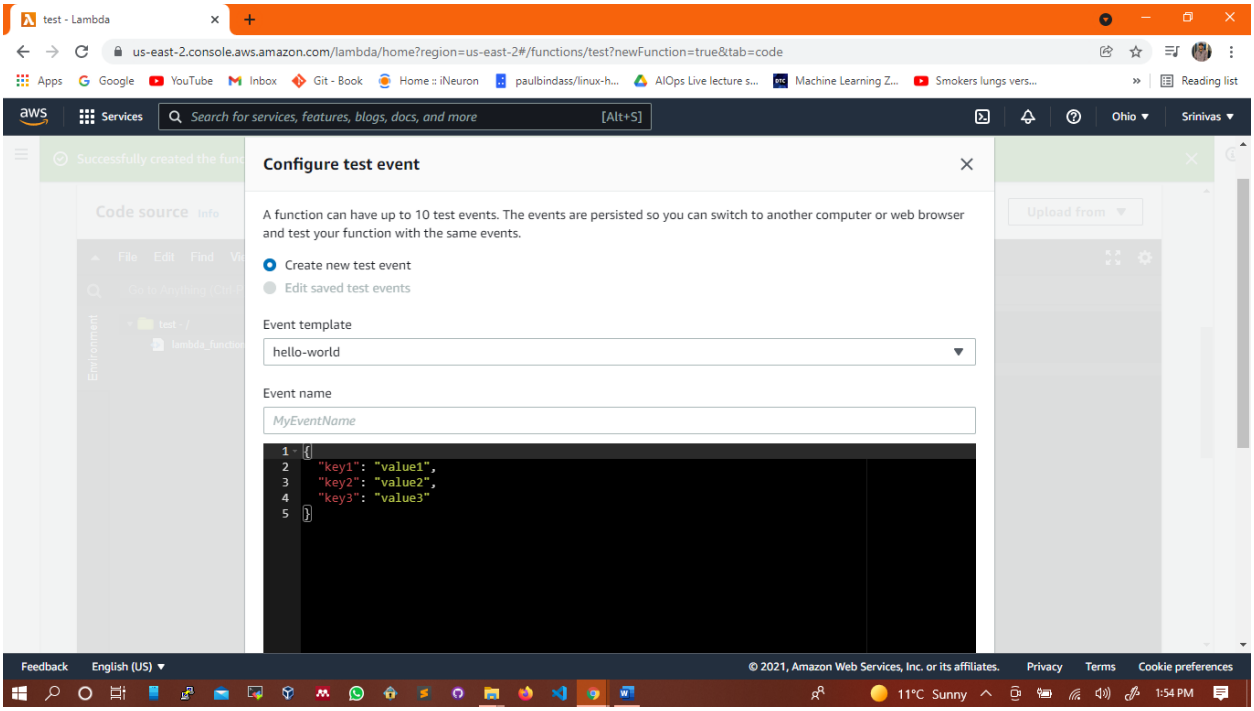
Parameters in the Lambda function –

[1] Event – whatever we want to pass to the lambda function we pass it here.

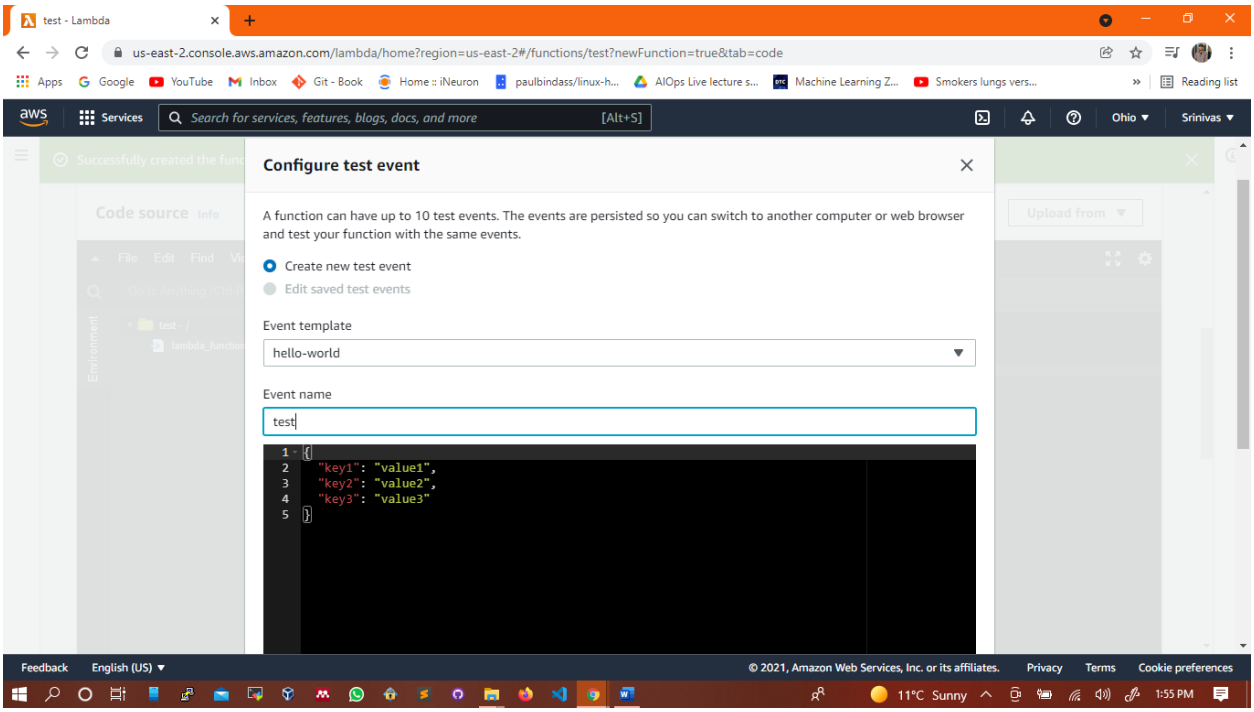
[2] context – None



Print the event parameter here.

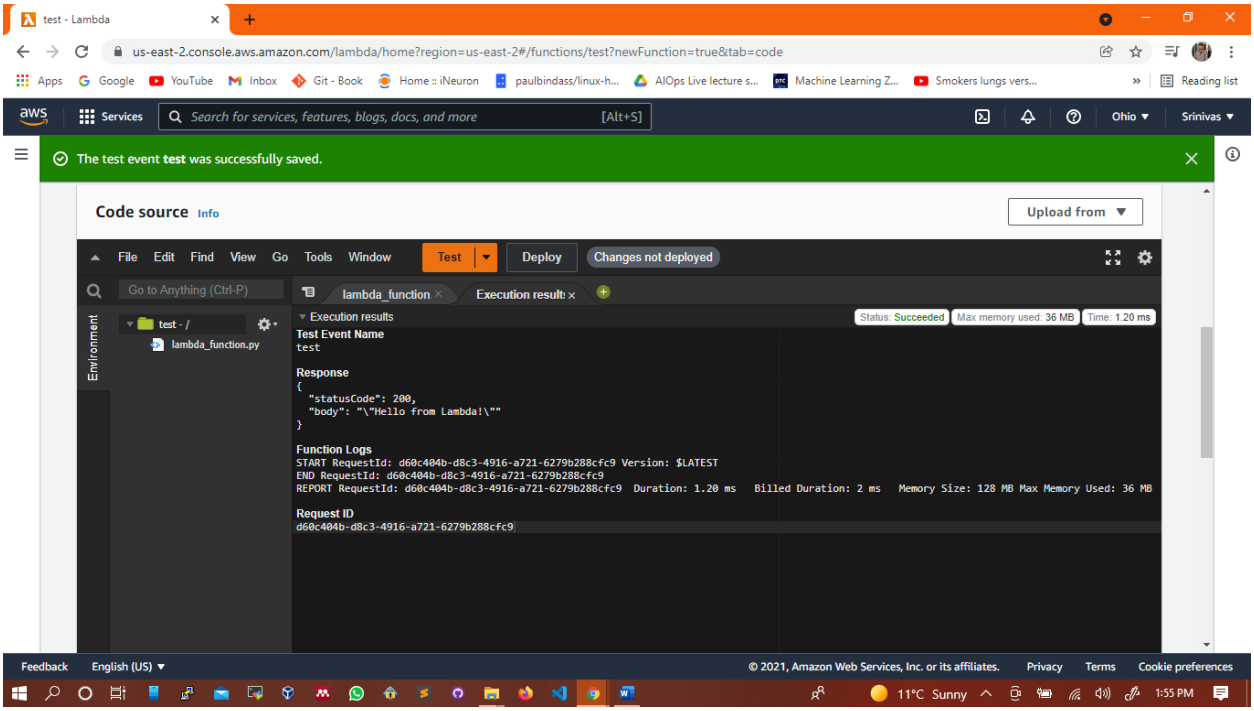


Here we get the test event. Whatever we want to test we pass it to the event parameter.

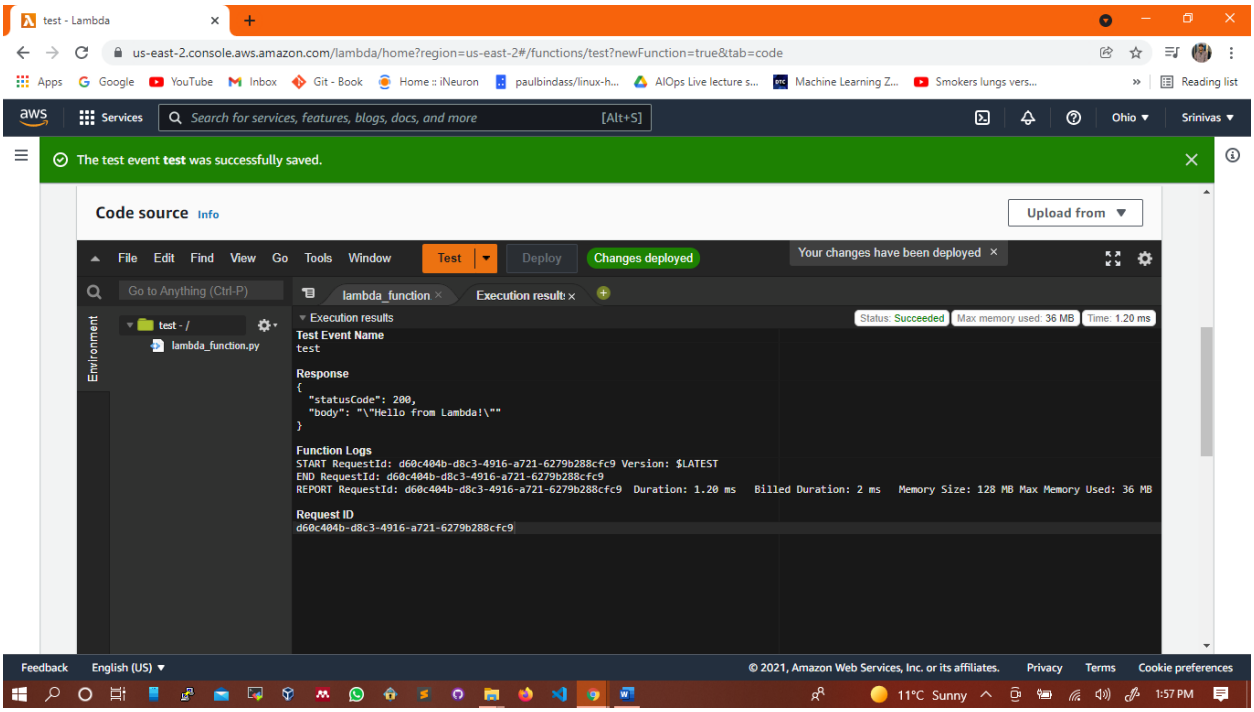


Creating the new test event.

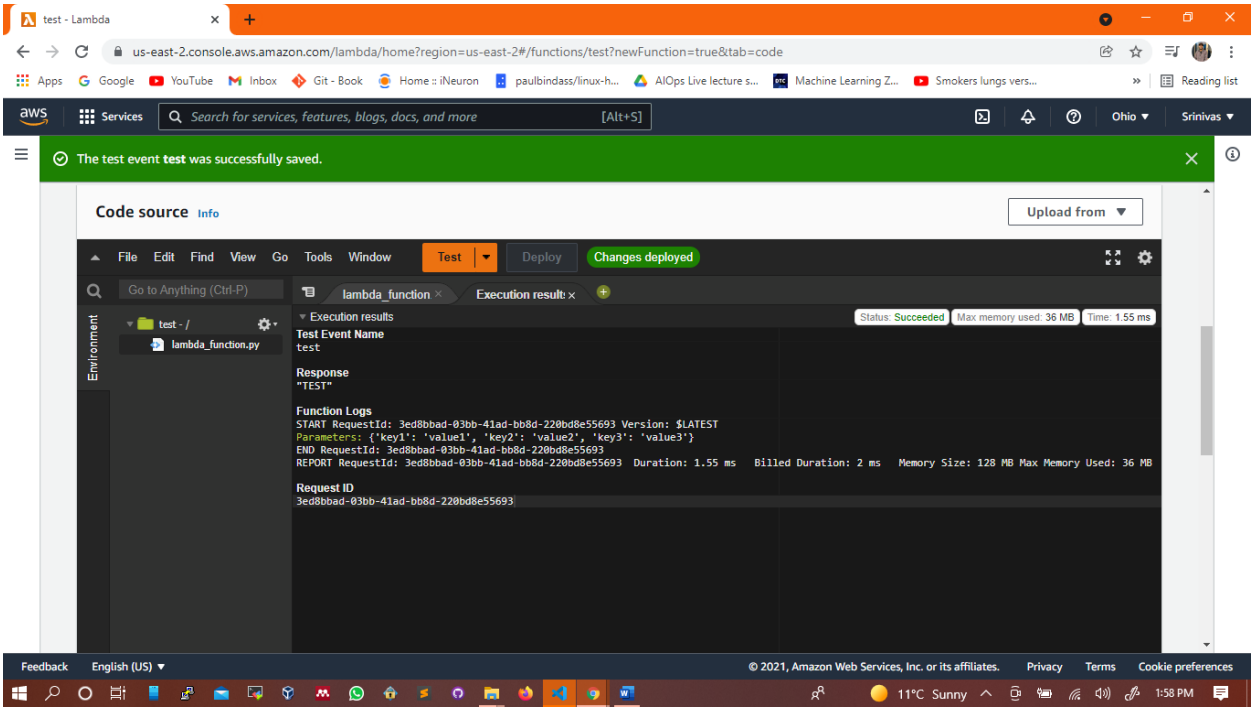
Testing lambda function



Changes are not deployed here we need to update the lambda function. Click on Deploy.



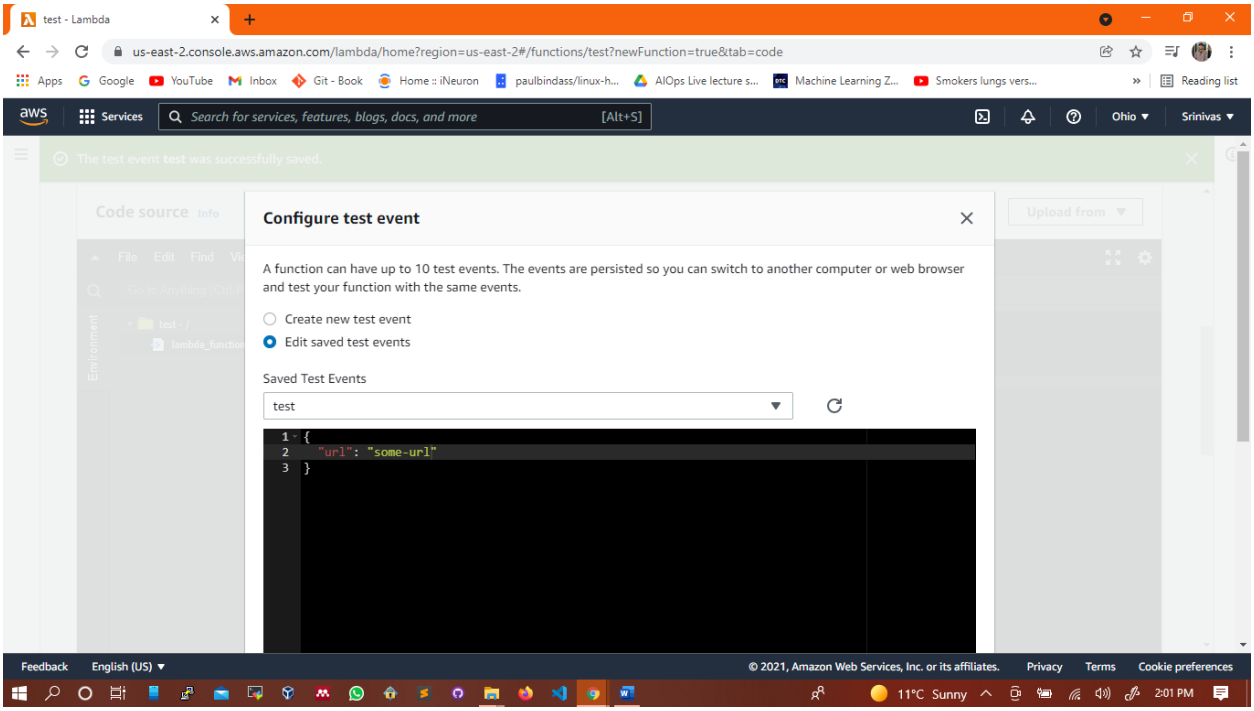
Then click Test

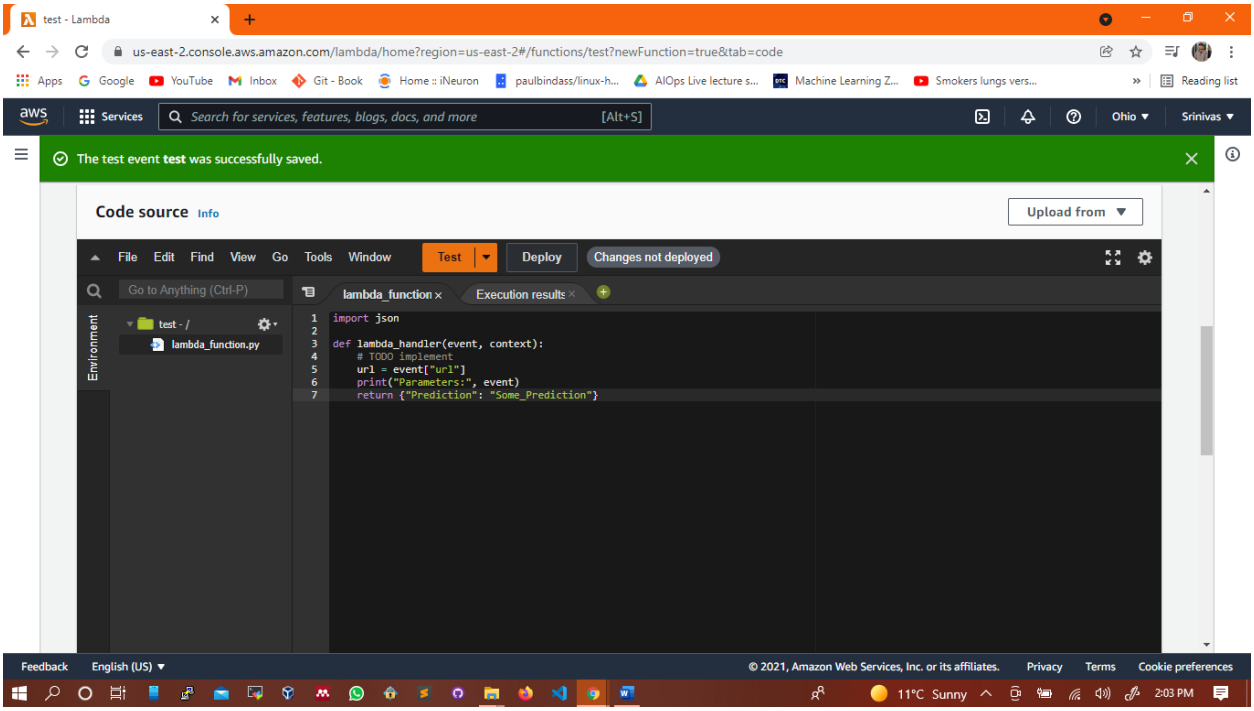


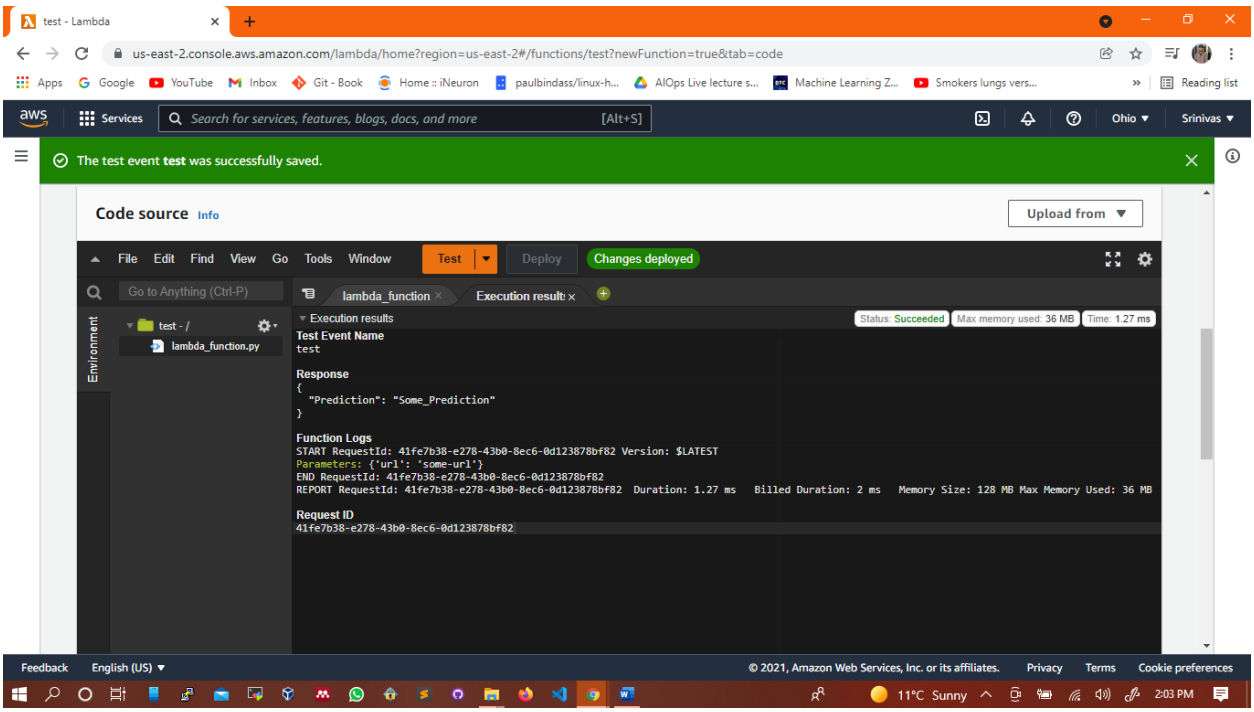
Here we see the parameters we pass and the response.

Here we pass a URL parameter to the lambda function.

ACCESSING THE URL



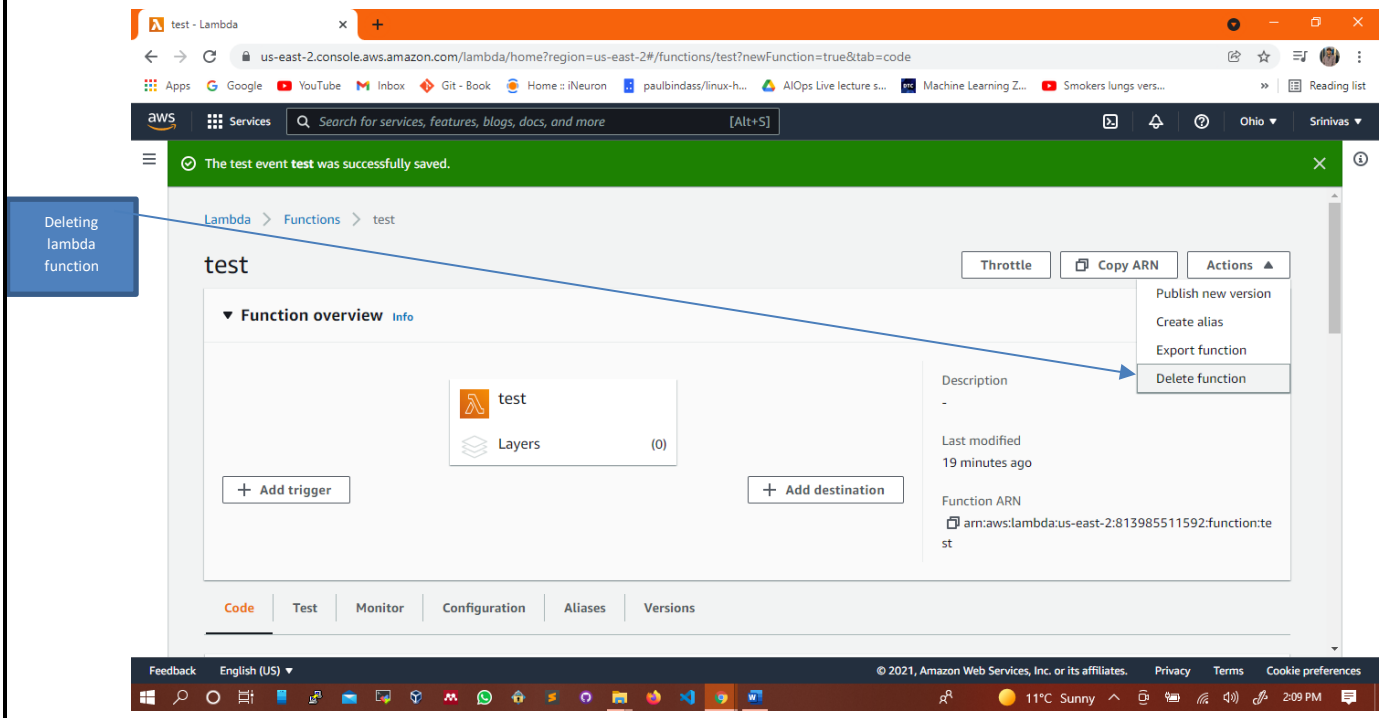

test - Lambda
us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions/test?newFunction=true&tab=code
Apps Google YouTube Inbox Git - Book Home :: iNeuron paulbindass/linux-h... AIOps Live lecture s... Machine Learning Z... Smokers lungs vers... Reading list
aws Services Search for services, features, blogs, docs, and more [Alt+S] Ohio Srinivas
The test event test was successfully saved.
Code source Info
Upload from
File Edit Find View Go Tools Window Test Deploy Changes not deployed
Go to Anything (Ctrl-P)
Environment
test - /
lambda_function.py
lambda_function Execution results
import json
def lambda_handler(event, context):
# TODO implement
url = event["url"]
print("Parameters:", event)
return {"Prediction": "Some_Prediction"}
Feedback English (US)
© 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
11°C Sunny 2:03 PM

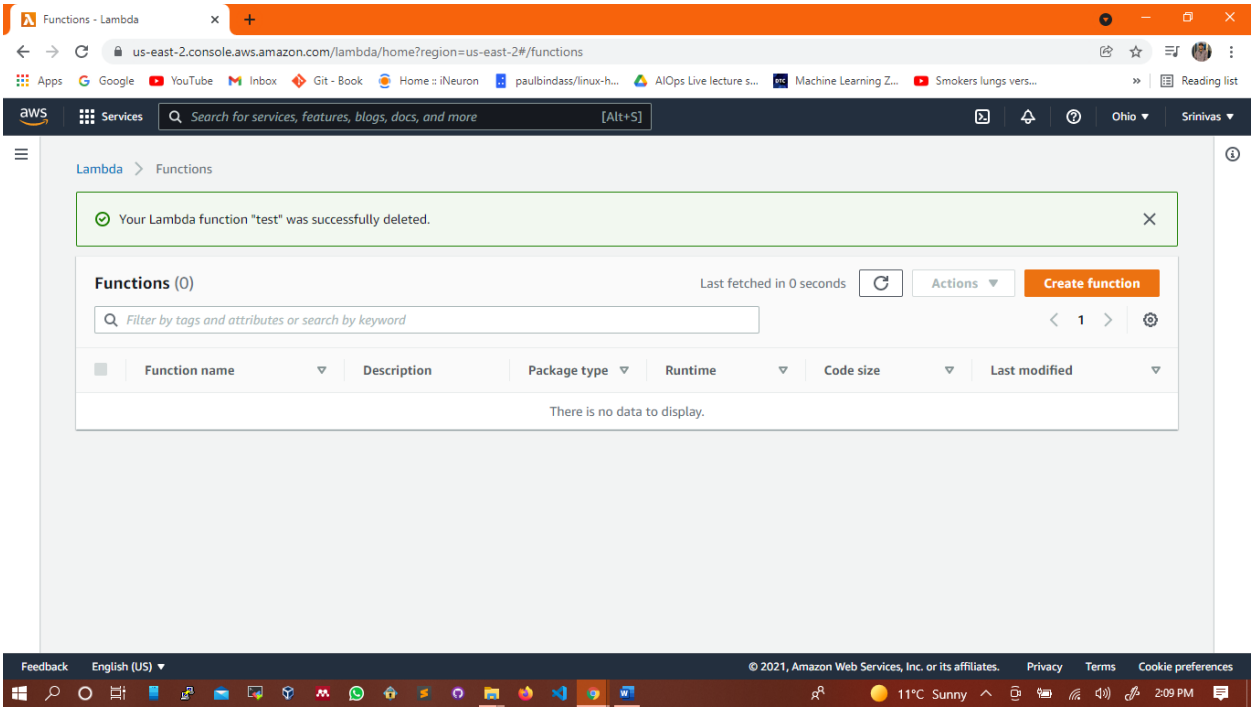

test - Lambda
us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions/test?newFunction=true&tab=code
Apps Google YouTube Inbox Git - Book Home :: iNeuron paulbindass/linux-h... AIOps Live lecture s... Machine Learning Z... Smokers lungs vers... Reading list
aws Services Search for services, features, blogs, docs, and more [Alt+S] Ohio Srinivas
The test event test was successfully saved.
Code source Info
Upload from
File Edit Find View Go Tools Window Test Deploy Changes deployed
Go to Anything (Ctrl-P)
Environment
test - /
lambda_function.py
lambda_function Execution result
Execution results
Status: Succeeded Max memory used: 36 MB Time: 1.27 ms
Test Event Name
test
Response
{
"Prediction": "Some_Prediction"
}
Function Logs
START RequestId: 41fe7b38-e278-43b0-8ec6-0d123878bf82 Version: $LATEST
Parameters: {'url': 'some-url'}
END RequestId: 41fe7b38-e278-43b0-8ec6-0d123878bf82
REPORT RequestId: 41fe7b38-e278-43b0-8ec6-0d123878bf82 Duration: 1.27 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 36 MB
Request ID
41fe7b38-e278-43b0-8ec6-0d123878bf82
Feedback English (US)
© 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
11°C Sunny 2:03 PM

Advantages of using lambda function

There is no need to create any EC2 instances and no need to think about any servers. Here we just need to write a function and just user it.

No need to think about the infrastructure for serving the URL for serving the models. Here we pay only when lambda function is responding and don't pay when it is idol and not in use. Here it is serverless and we don't need to worry about servers.





Function is deleted.

TENSORFLOW LITE

AWS Lambda has limit of the file size if can serve. That is less than 50MB zip file.

At present we have docker and have a larger limit. Up to 10GB. But if we have a larger image size it will be difficult to store the image. After invoking the Lambda function, it will take some time to initialize. Also initializing the tensorflow also takes lot of time. And it also takes more RAM in it.

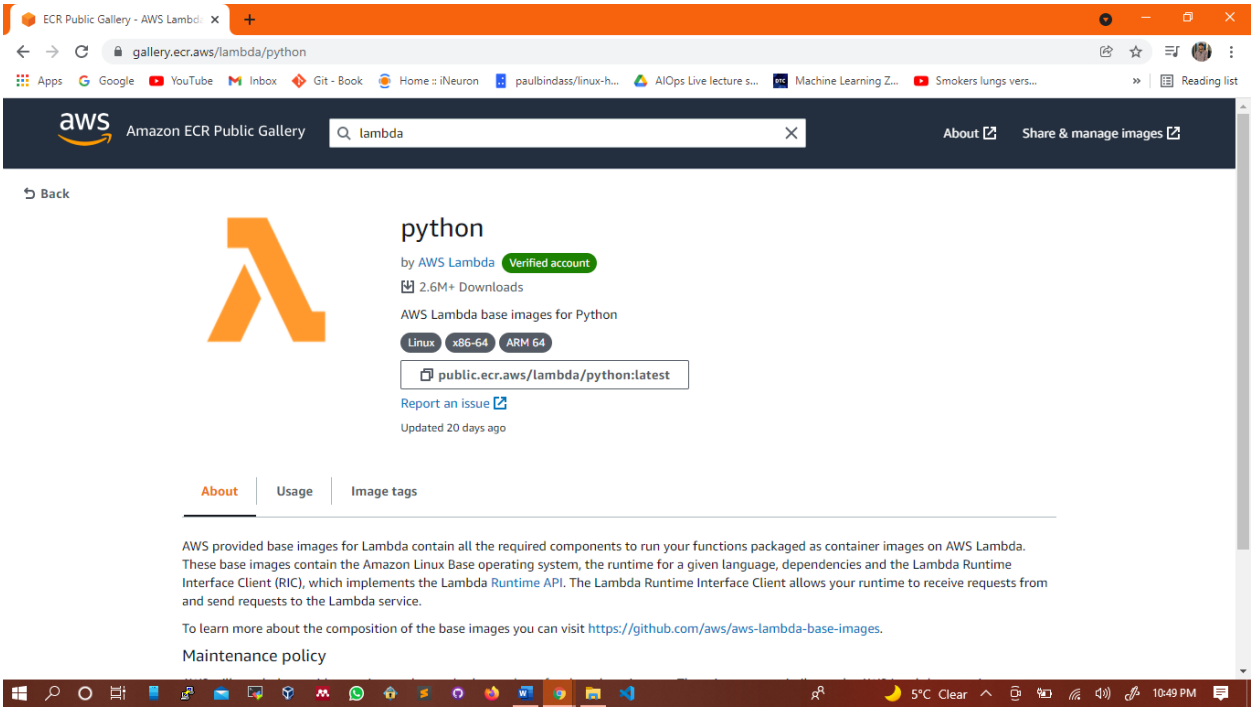
Here we use a lighter version of tensorflow which is tensorflow-lite. This will only focus on inference. It is not useful for training the models but only used for inferencing the models.

PREPARING CODE FOR THE LAMDA

IMAGE – TESTING THE IMAGE

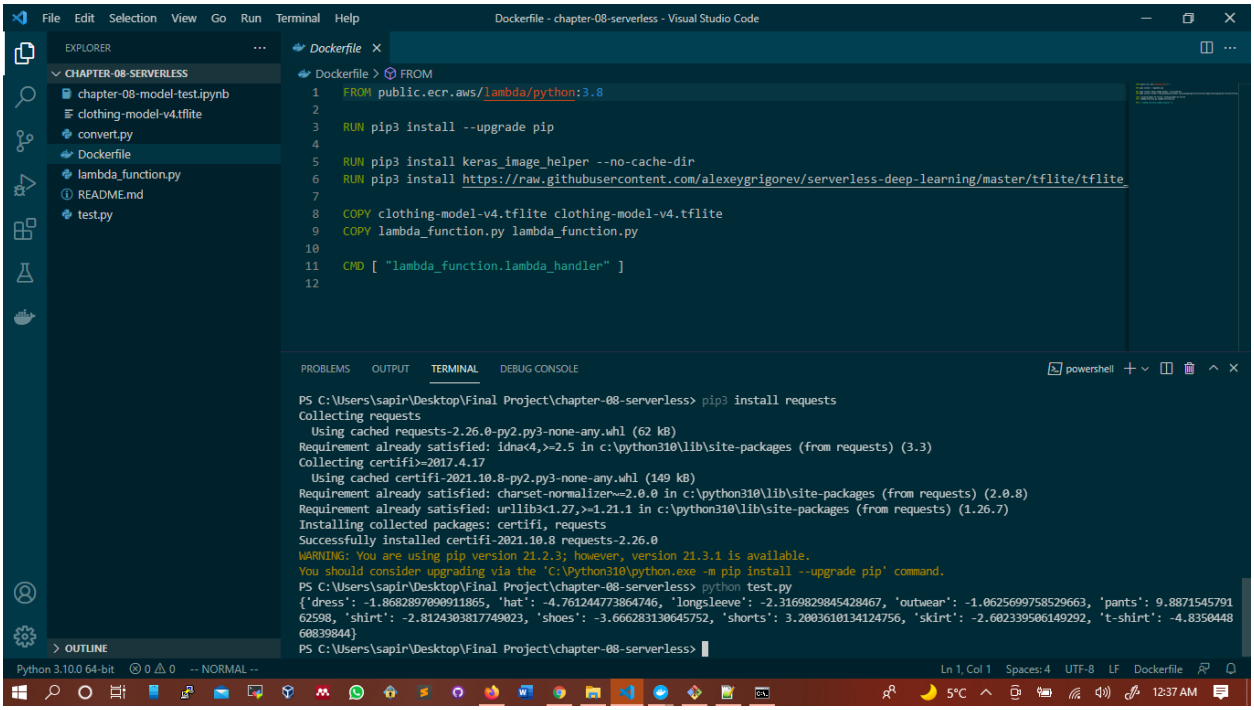
We prepare docker from base images prepared from AWS.

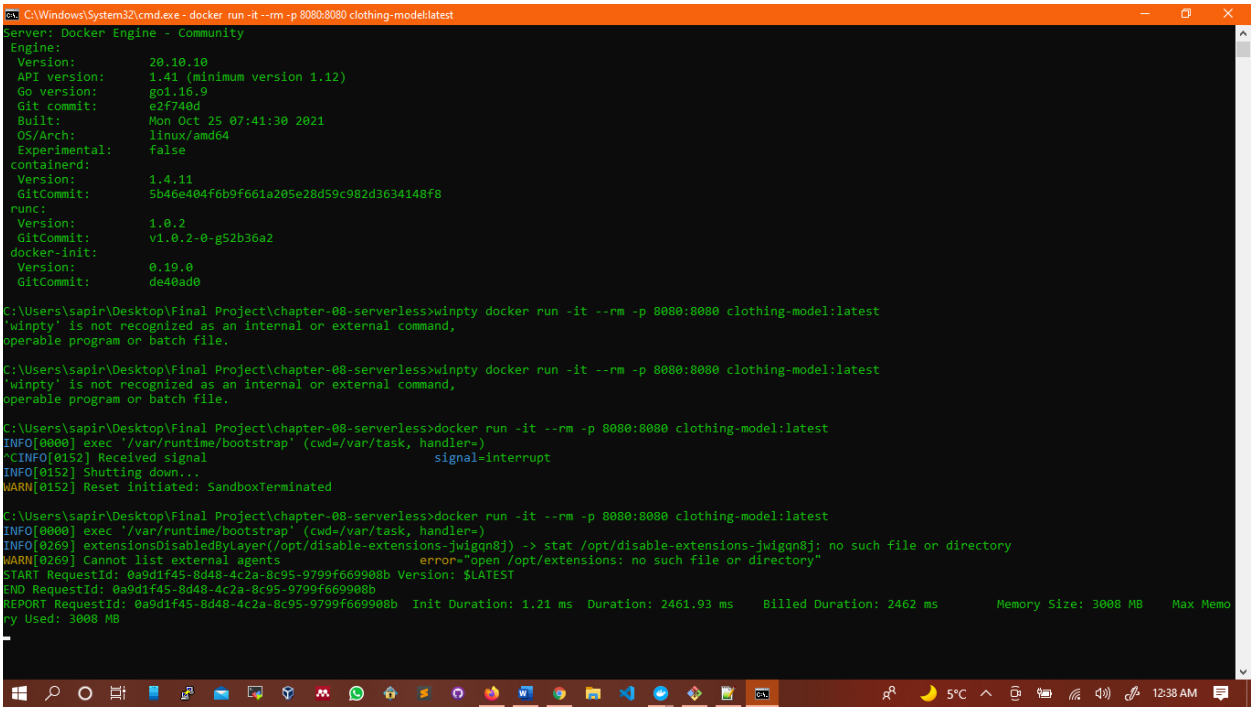
https://gallery.ecr.aws/lambda/python



Docker build -t clothing-model .

winpty docker run -it –-rm -p 8080:8080 clothing-model:latest





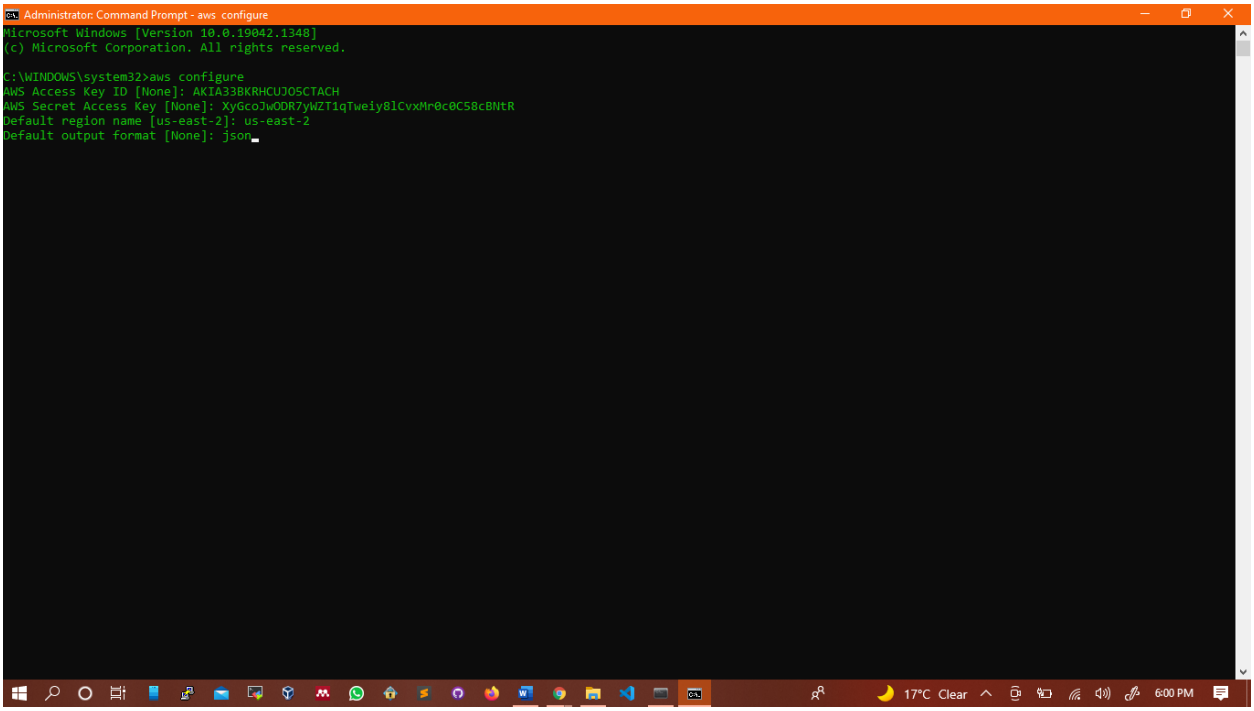
CREATING LAMBDA FUNCTION

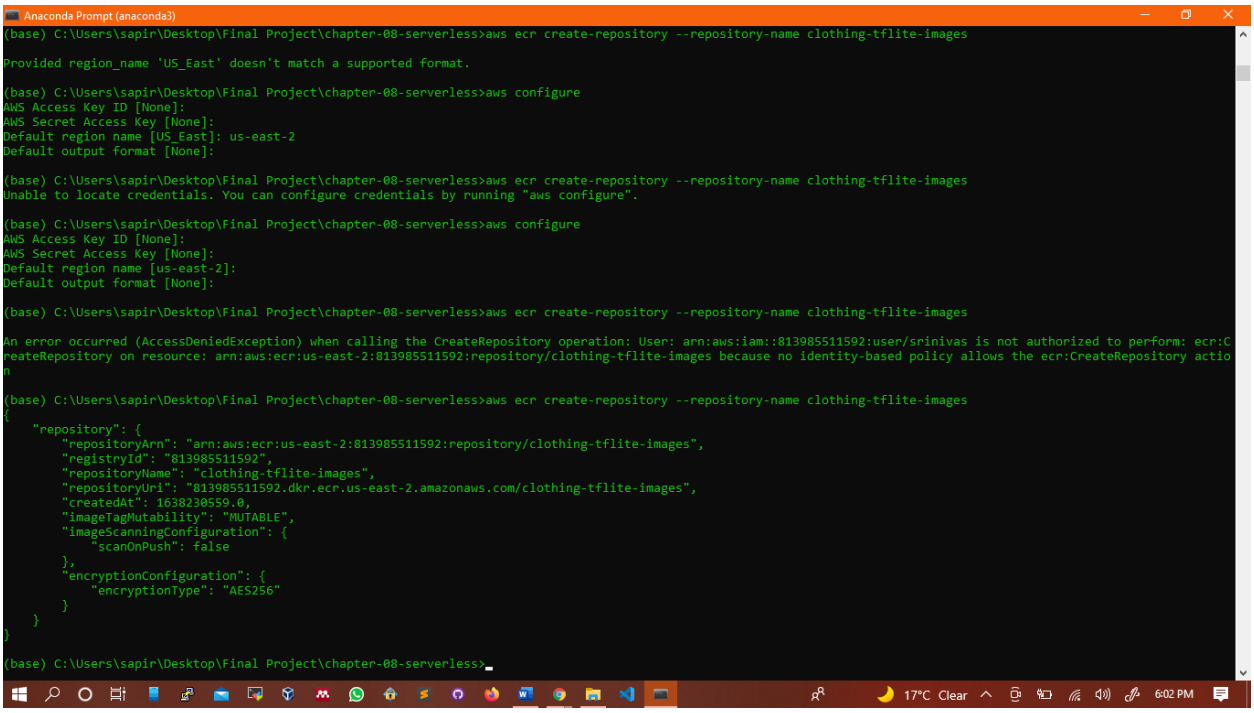
Here we need to deploy the docker image to lambda. We publish the docker images in AWS ECR. ECR is where we publish the docker images.

ECR – Elastic Container Repository

pip install awscli

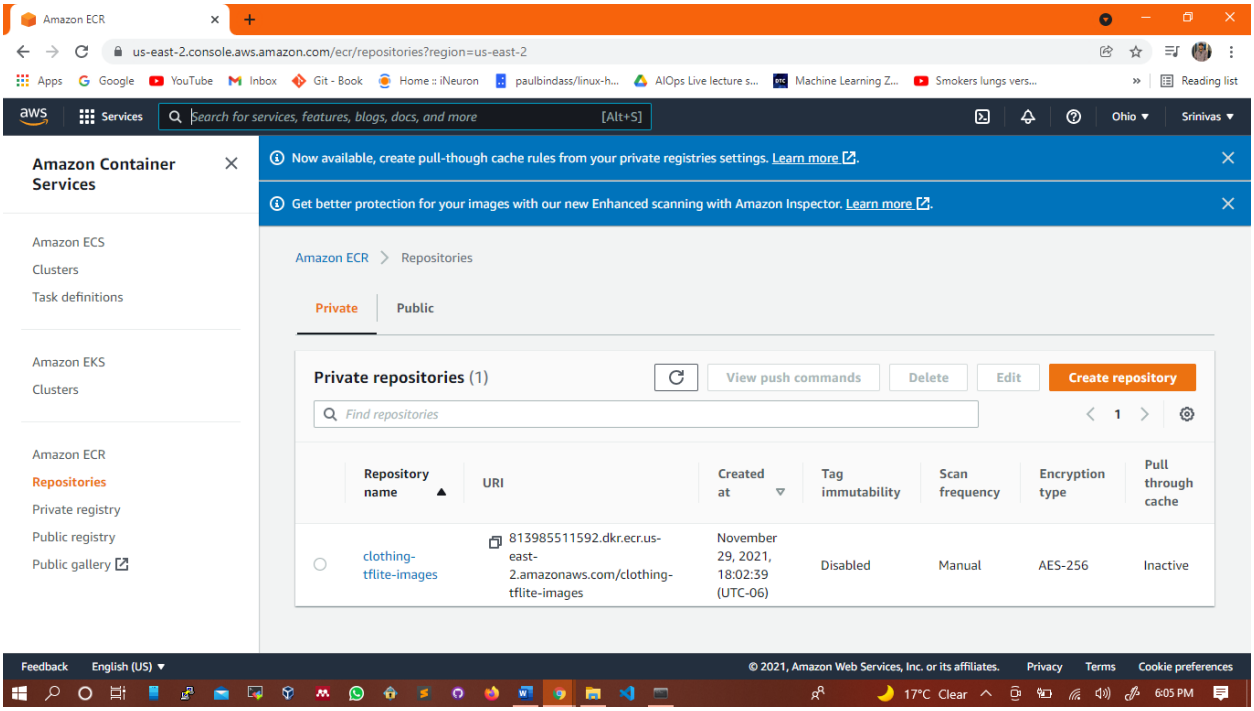
aws ecr create-repository –repository-name clothing-tflite-images





"repositoryUri": "813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images"

AWS ECR Container Service



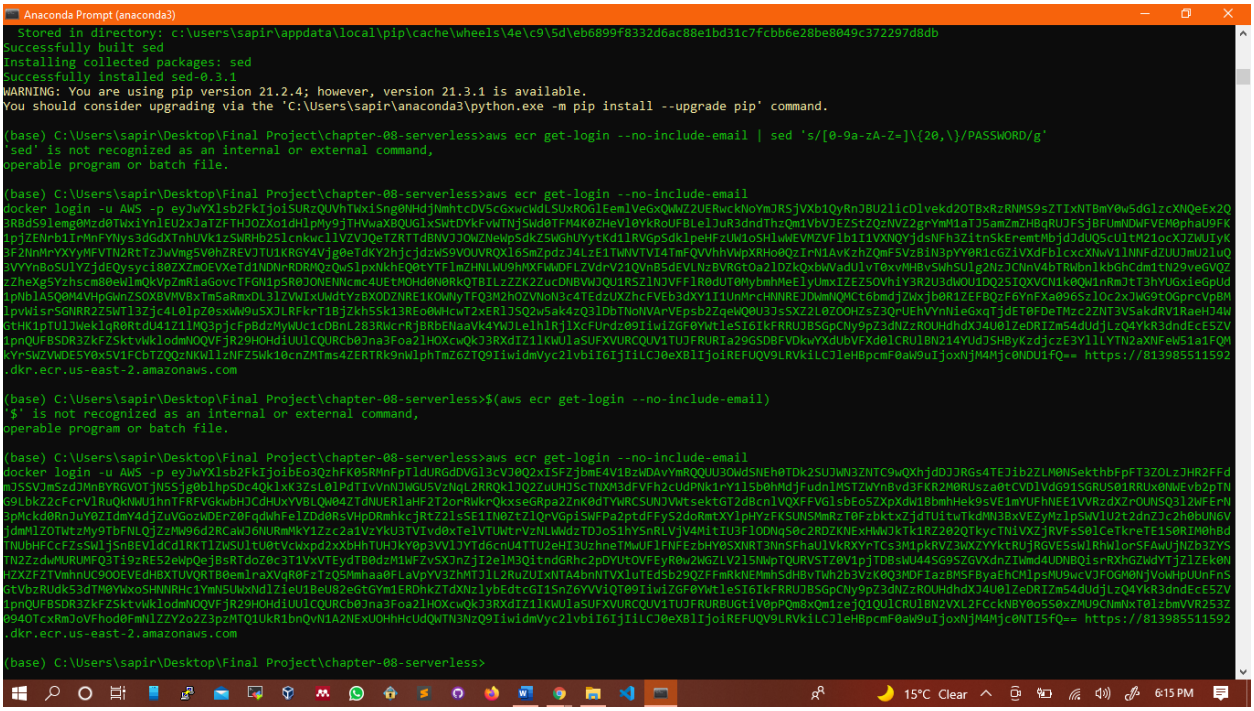
Then we need to publish the image we have created.

We need to login to the registry to push there.

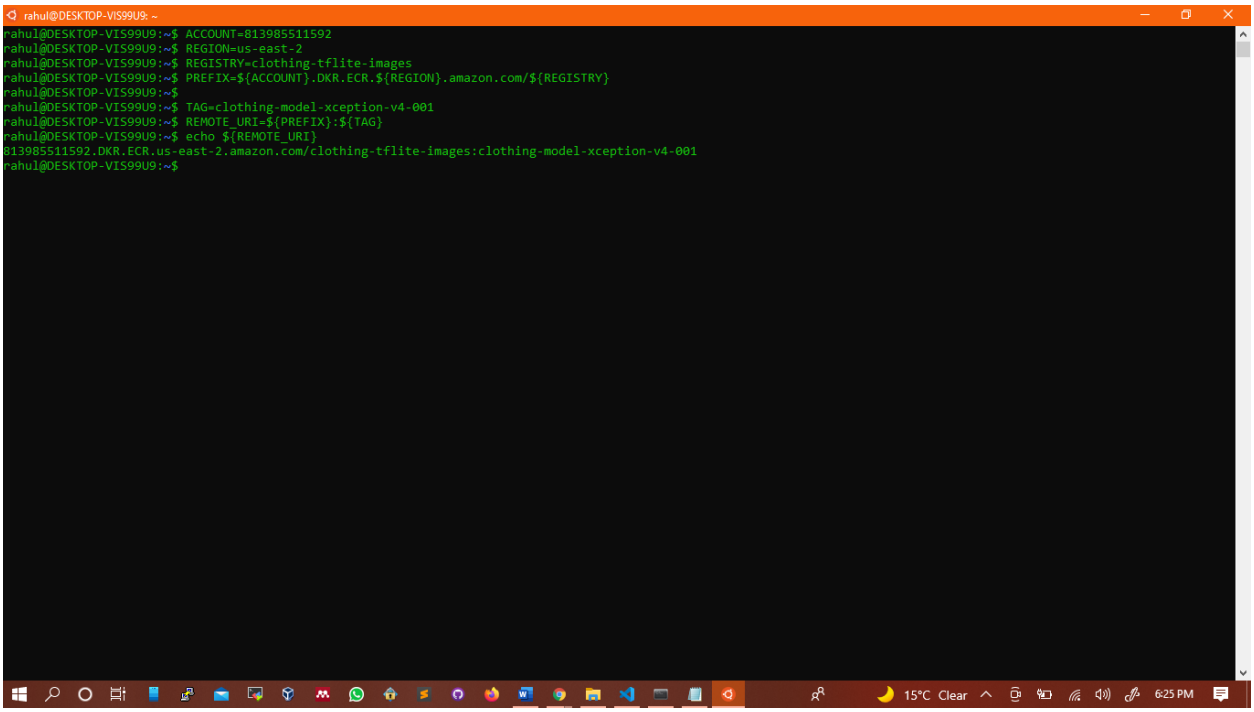
aws ecr get-login –-no-include-email | sed ‘s/[0-9a-zA-Z=]\{20,\}/PASSWORD /g’

The output of this will help to login to the registory and we will be able to push the registory.

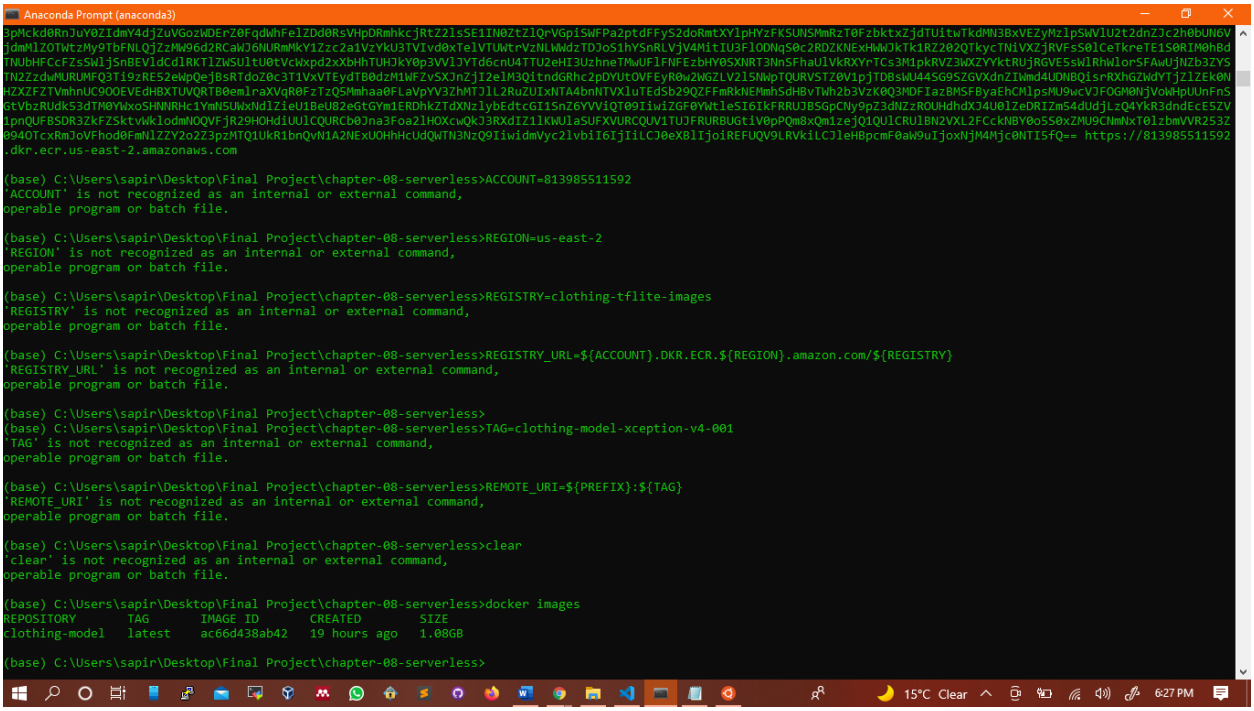
$(aws ecr get-login –-no-include-email)



https://813985511592.dkr.ecr.us-east-2.amazonaws.com

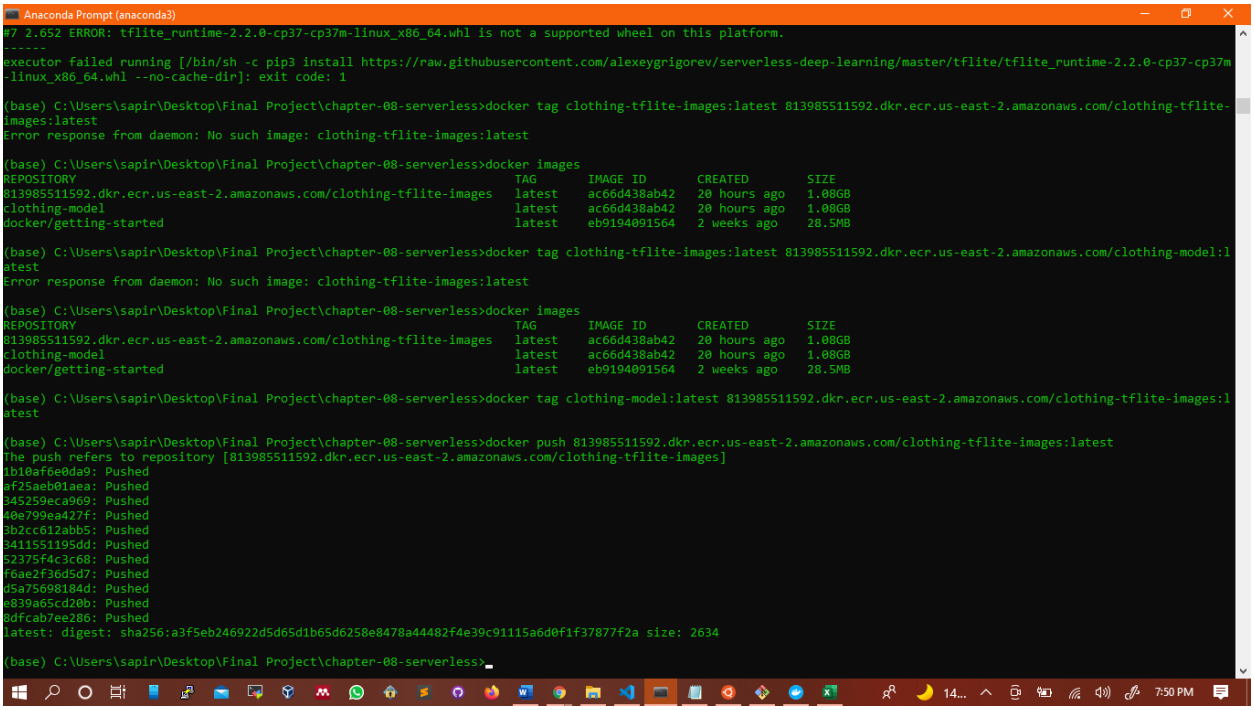


813985511592.DKR.ECR.us-east-2.amazon.com/clothing-tflite-images:clothing-model-xception-v4-001



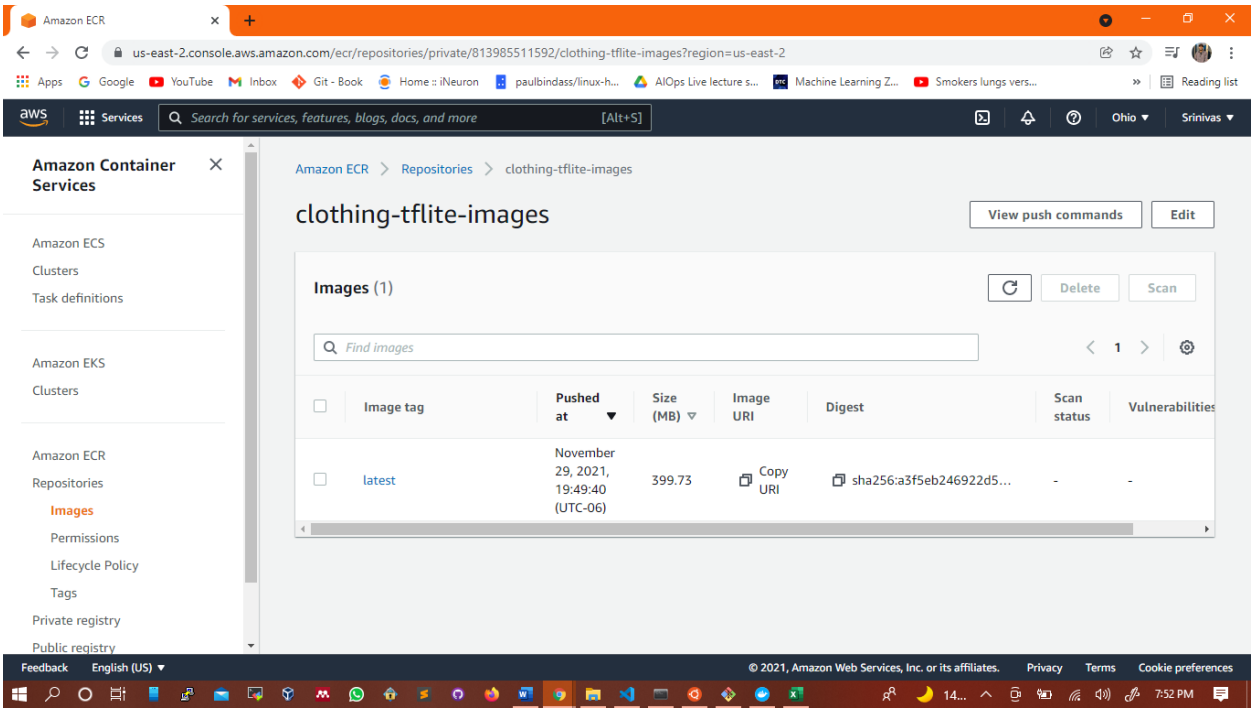
Tag this image with the URI that we have just created.

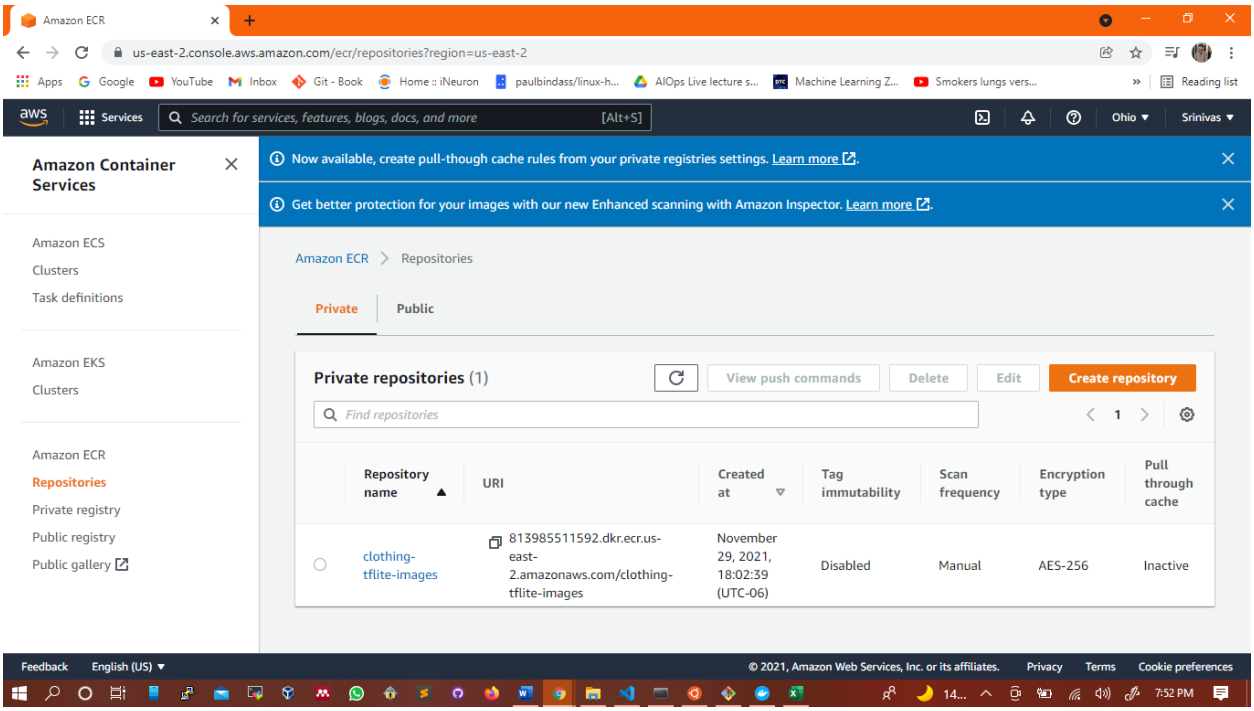
docker tag clothing-model:latest ${REMOTE_URI}



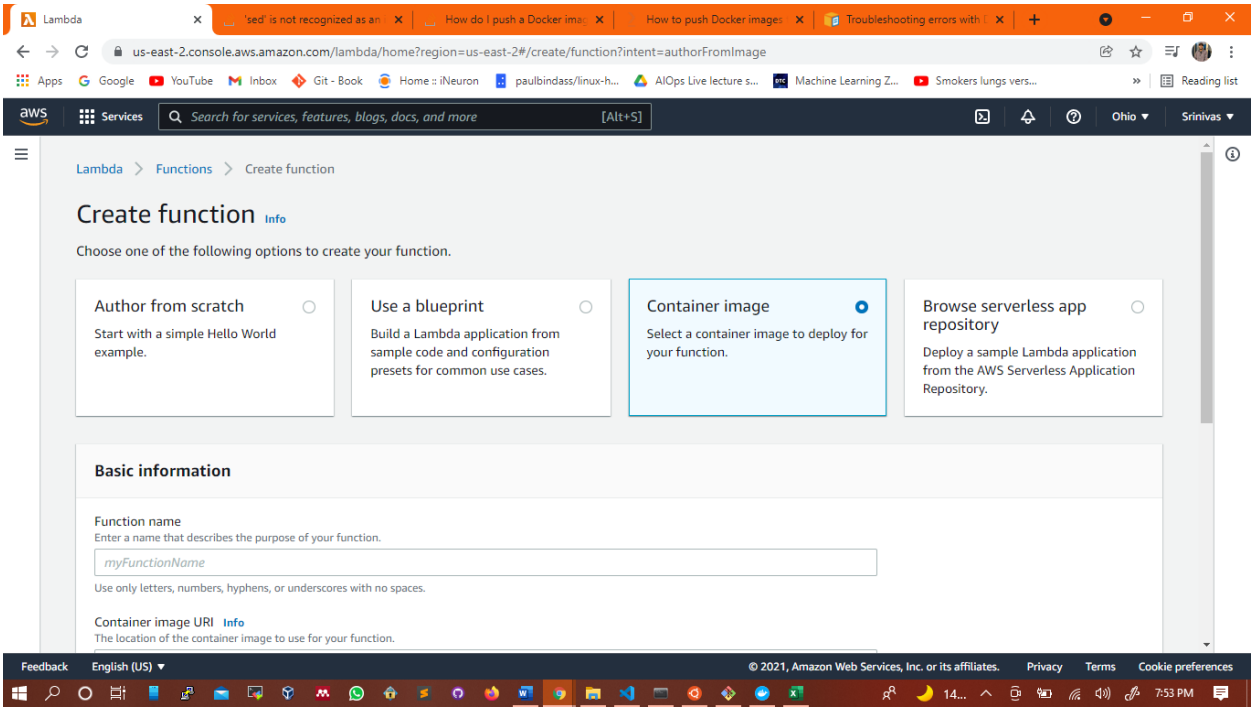
docker tag clothing-model:latest 813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images:latest

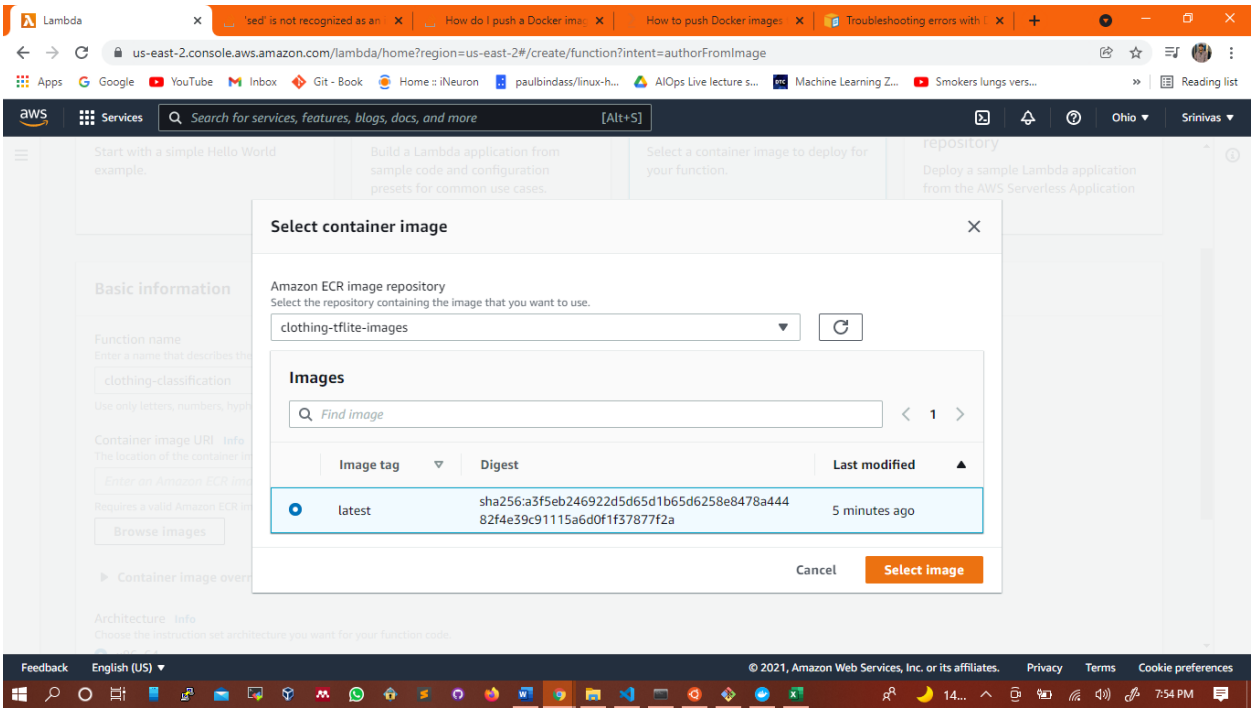
docker push 813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images:latest

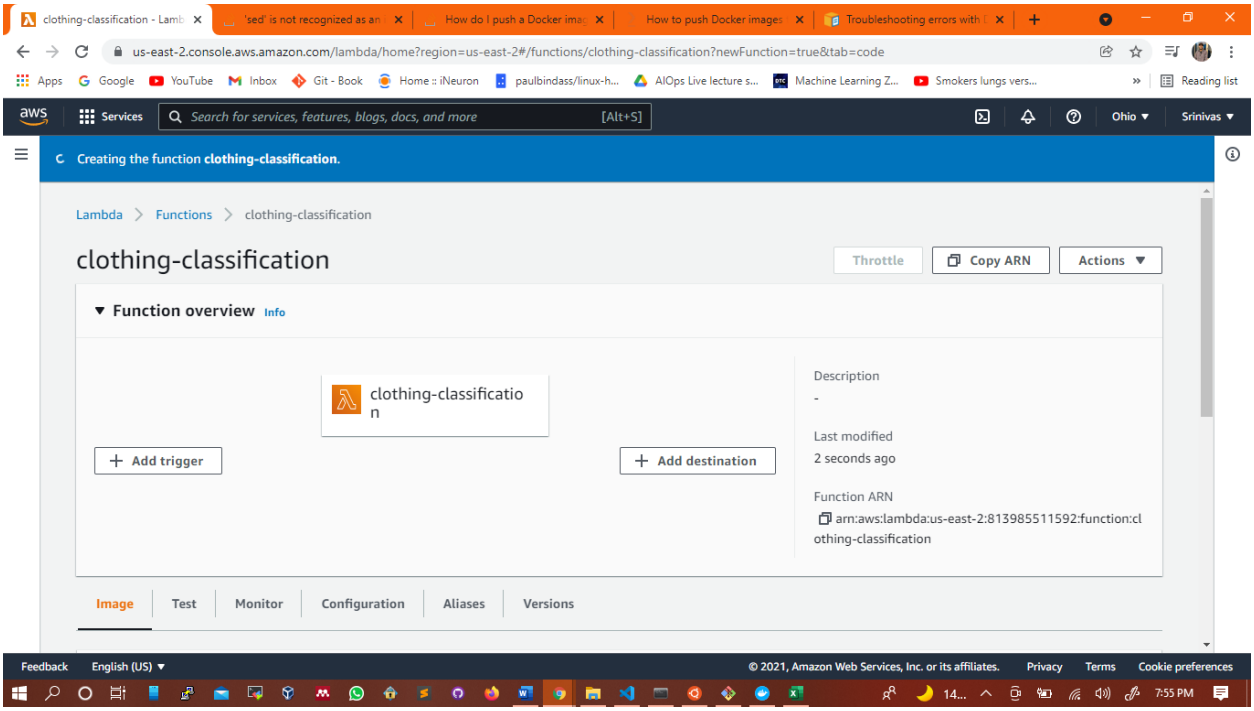




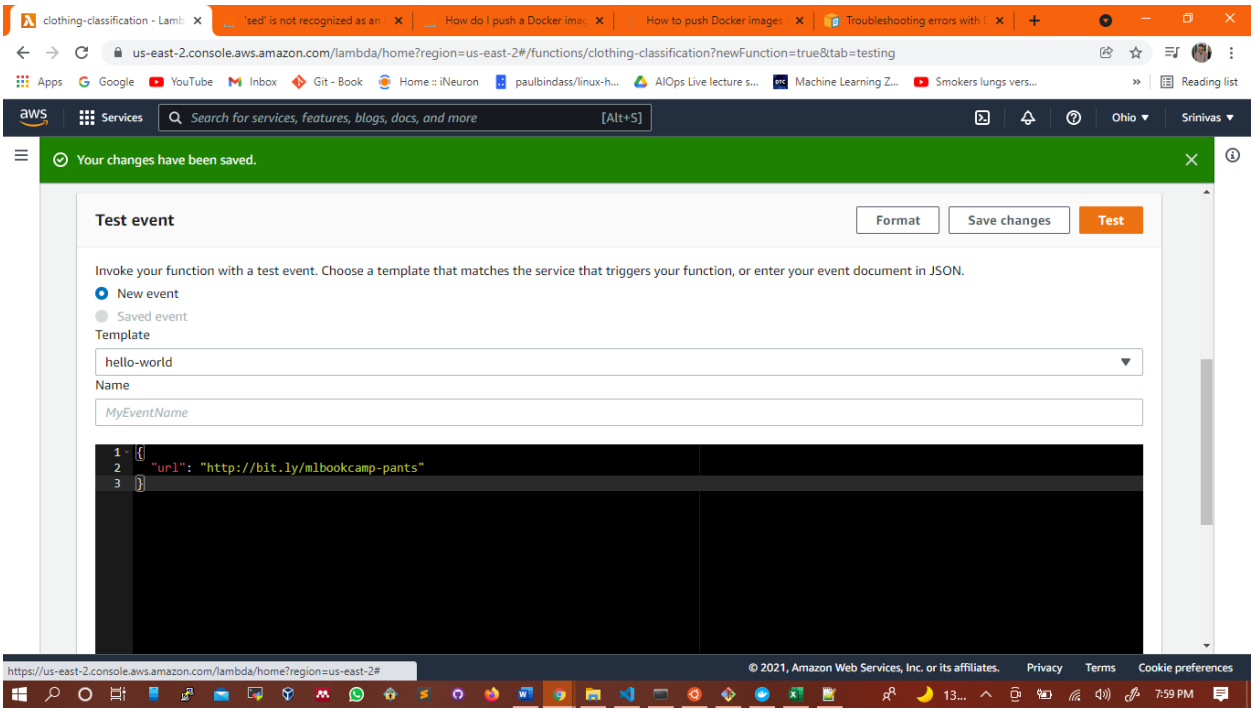
Creating ECR Container Image

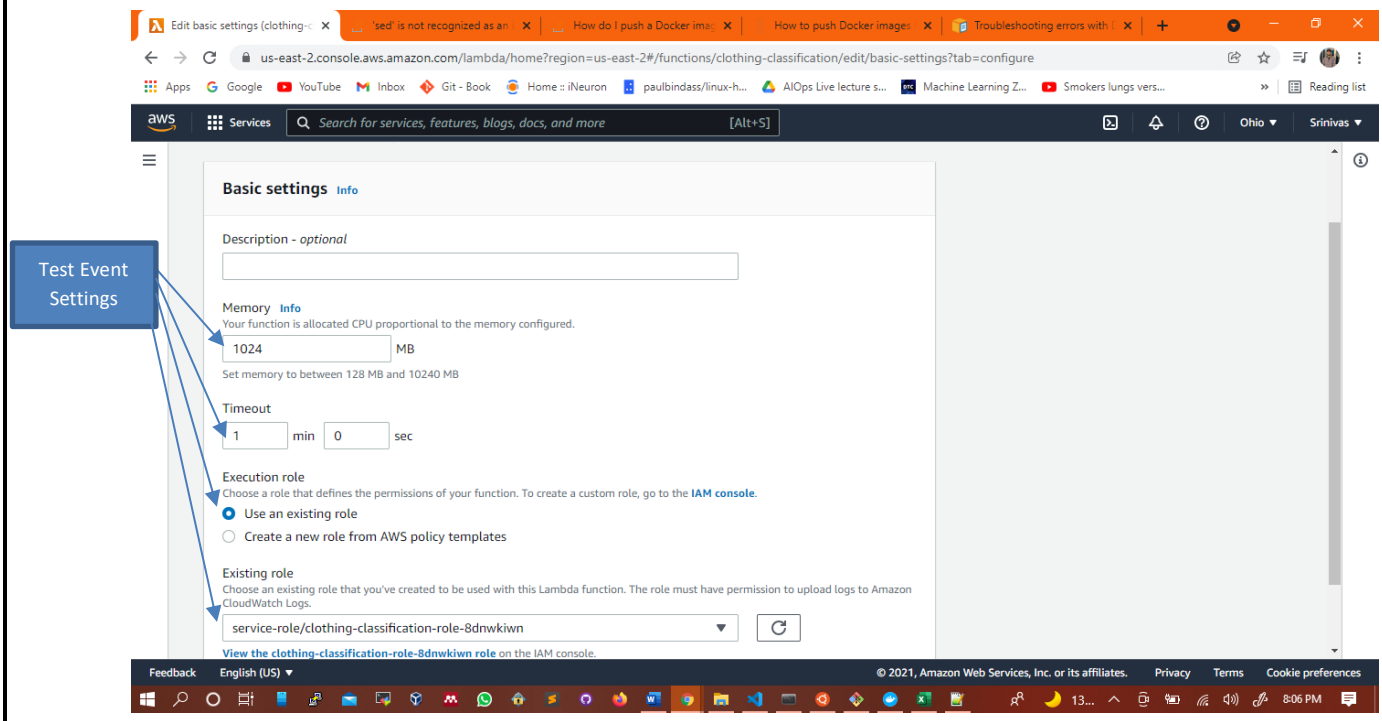


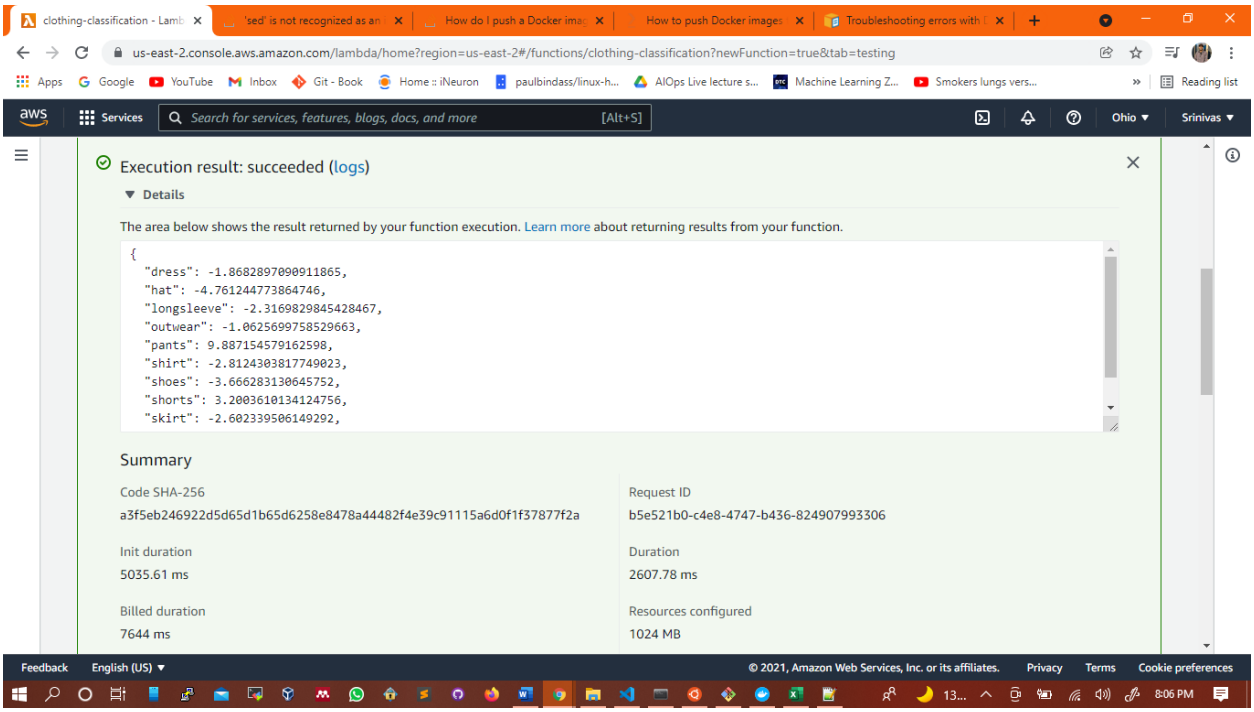




INVOKING THE FUNCTION WITH THE TEST EVENT – increase the timeout to 30 seconds




Test Event Settings
Edit basic settings (clothing-c
us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions/clothing-classification/edit/basic-settings?tab=configure
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ohio
Srinivas
Basic settings Info
Description - optional
Memory Info
Your function is allocated CPU proportional to the memory configured.
1024
MB
Set memory to between 128 MB and 10240 MB
Timeout
1
min
0
sec
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Use an existing role
Create a new role from AWS policy templates
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
service-role/clothing-classification-role-8dnwkiwn
View the clothing-classification-role-8dnwkiwn role on the IAM console.
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
8:06 PM


clothing-classification - Lamb
us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions/clothing-classification?newFunction=true&tab=testing
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ohio
Srinivas
Execution result: succeeded (logs)
Details
The area below shows the result returned by your function execution. Learn more about returning results from your function.
{
"dress": -1.8682897090911865,
"hat": -4.761244773864746,
"longsleeve": -2.3169829845428467,
"outwear": -1.0625699758529663,
"pants": 9.887154579162598,
"shirt": -2.8124303817749023,
"shoes": -3.666283130645752,
"shorts": 3.2003610134124756,
"skirt": -2.602339506149292,
Summary
Code SHA-256
a3f5eb246922d5d65d1b65d6258e8478a44482f4e39c91115a6d0f1f37877f2a
Request ID
b5e521b0-c4e8-4747-b436-824907993306
Init duration
5035.61 ms
Duration
2607.78 ms
Billed duration
7644 ms
Resources configured
1024 MB
Feedback
English (US)
© 2021, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
8:06 PM

Future Work

- Understanding different methods of deploying and serving models in the cloud.
- Serving Keras and TensorFlow models with TensorFlow-Serving
- Deploying TensorFlow-Serving to Kubernetes